JavaScript - Perusteet 1

Tässä materiaalissa ei ole lueteltu JavaScriptin ominaisuuksia käsikirjan tapaan. Esitys pyrkii olemaan tiivis ottaen huomioon, että lukijalla on olemassa perustiedot HTML:stä, CSS:stä ja ohjelmoinnin perusteista (muuttujat, valinta- ja toistorakenteet jne.)

Tätä materiaalia vastaavaa sisältöä löytyy mm. seuraavista

- MDN - JavaScript basics
- http://www.w3schools.com/js/

JavaScript-ohjelmointikieli

- HTML:n yhteydessä pääasiassa **selainkriptien** tekemiseen käytetty kieli
- Ohjelmakoodia, jonka selain suorittaa HTML-dokumentin näyttämisen yhteydessä
- Suorittajana selaimen JavaScript-tulkki eli JavaScript-moottori (mm. Googlen V8 tai Firefoxin IonMonkey)
- JavaScript-ohjelmointikieli on standardoitu ECMAScript-kielenä
- Pohjana Netscapen 1995 kehittämä skriptikieli (aluksi LiveScript)
- JavaScript ei ole sidoksissa HTML:ään, mutta useimmin sillä käsitellään HTML-dokumentteja

JavaScriptillä voidaan

- muokata HTML-dokumentin rakennetta ja sisältöä
- muokata HTML-dokumentin ulkoasua
- luoda vuorovaikutteisuutta ja toiminnallisuutta
- laajimmillaan muodostaa HTML5-sovellus, jolloin HTML:n osuus minimoituu script-elementin esittämiseen

EcmaScript vs. JavaScript

- EcmaScript on varsinainen kielen standardi
  - Eri versioita, lyhennetään yleisesti: ... ES5, **ES6**, ES7, ES8.
- JavaScript (selaimessa oleva tulkki) on murre, joka seuraa tiettyä ES:n versiota joko kokonaan tai osittain mahdollisine omine lisäyksineen
- Tässä materiaalissa pyritään noudattamaan ES6-standardia sisältäen monien aiempien ES-versioiden ominaisuuksia

- Tällä hetkellä (12.5.2018) ES6 on varsin hyvin tuettu kaikissa keskeisissä selaimissa
- Pätevä kartta ES6-version selaintuesta

JavaScriptin liittäminen HTML-dokumenttiin

Kolme tapaa, joista ensimmäinen paras

- ulkoinen `<script src="skripti.js"></script>`
- script-elementti: `<script>alert('Haloo');</script>`
- tapahtumamäärite `<body onload="alert('Haloo')">`

- Ilman erityisvaatimuksia **script**-elementti kannattaa sijoittaa **body**-elementin viimeiseksi elementiksi. Tällöin ohjelmakoodilla mahdollisesti muokattava **HTML**-osio on jo kokonaisuudessaan selaimen muistissa
- Myös **head**-elementti on ok, jos yllä mainittia vaatimusta ei ole
- Script-elementtien esiintymiskertoja ole rajattu.

JavaScriptin liittäminen HTML-dokumenttiin

Esimerkki 0201

```
<!DOCTYPE html>
<html>
<head>
</head>

<body>
<script>document.write('Haloo Maailma!')</script>
</body>
</html>
```

Tulostaa "Haloo Maalima!"

Esimerkki 0202

```
<!DOCTYPE html>
<html>
<head>
<script>
function funktio() {
        document.getElementById("demo").innerHTML="My First JavaScript Function";
}
</script>
</head>

<body>

        <div id="demo"></div>
        <button type="button" onclick="funktio()">Try it</button>
</body>
</html>
```

Kokeile!

console.log

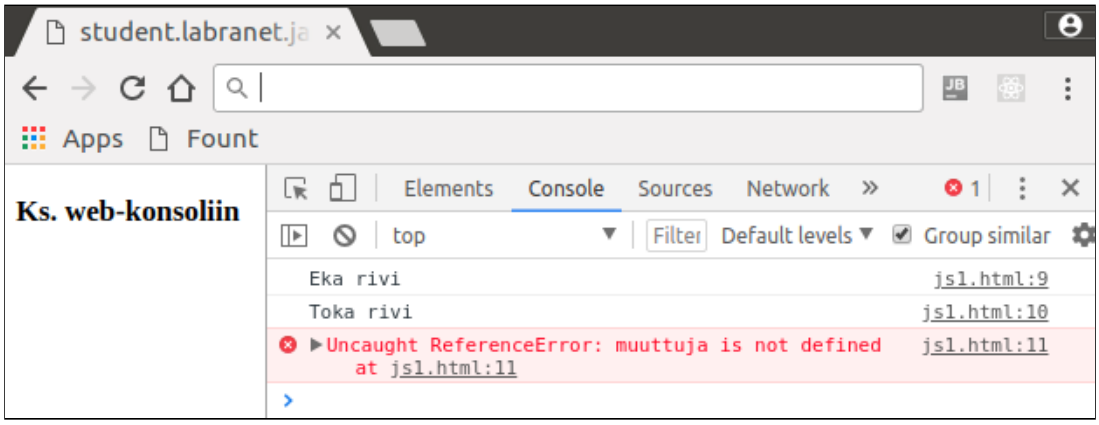
- Tulostaa selaimen web-konsoliin. Selaimen web-konsoli aktivoidaan yleensä F12-näppäimellä
- console.log()-metodi on käyttökelpoinen JavaScript-koodin testaamisessa ja debuggauksessa
- => Minimoidaan HTML/CSS-asioiden mukaanottaminen pelkän JavaScript-toiminnallisuuden kokeilemisessa ja testaamisessa

Esimerkki

```
<!DOCTYPE html>
<html>
<head>
</head>

<body>
<h3>Ks. web-konsoliin</h3>
<script>
console.log('Eka rivi');
console.log('Toka rivi');
console.log(muuttuja);
</script>
</body>
</html>
```

tulostaisi seuraavasti



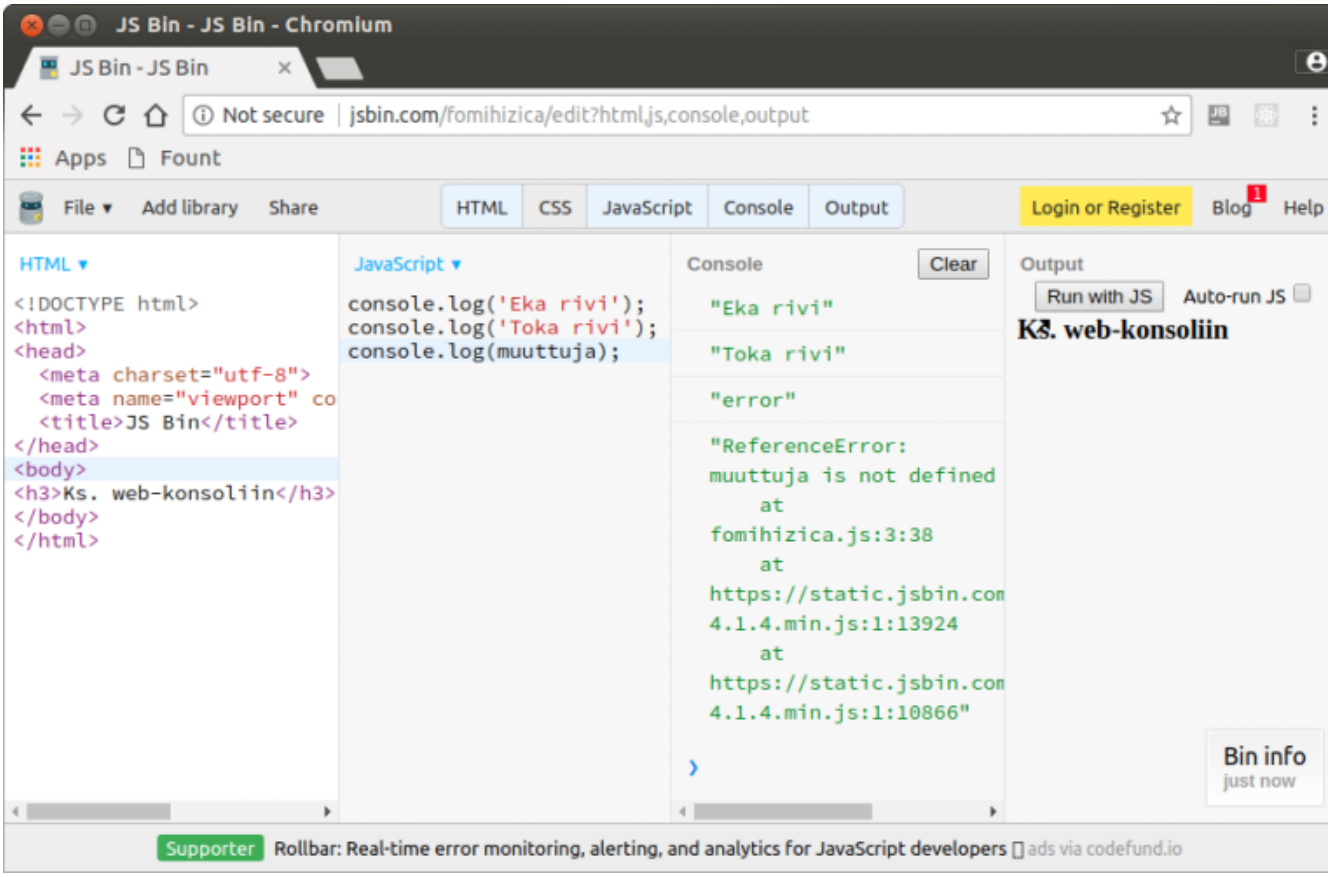
Huomaa määrittelemättömästä muuttujasta johtuva virheilmoitus

jsbin.com

- JavaScript-koodin ajaminen omasta paikallisesta tai palvelimen tiedostosta on helppoa, mutta opiskeluvaiheessa jatkuva muutosten tekeminen, tallentaminen ja ajaminen uudelleenlataamalla on turha "hidas" vaihe.
- Uuden asian omaksumisen ensimmäisissä kokeiluvaiheissa ohjelmakoodi on suoraviivaisinta suorittaa jossakin asiaan erikoistuneessa verkkopalvelussa.
- jsbin.com on yksi tällainen palvelu

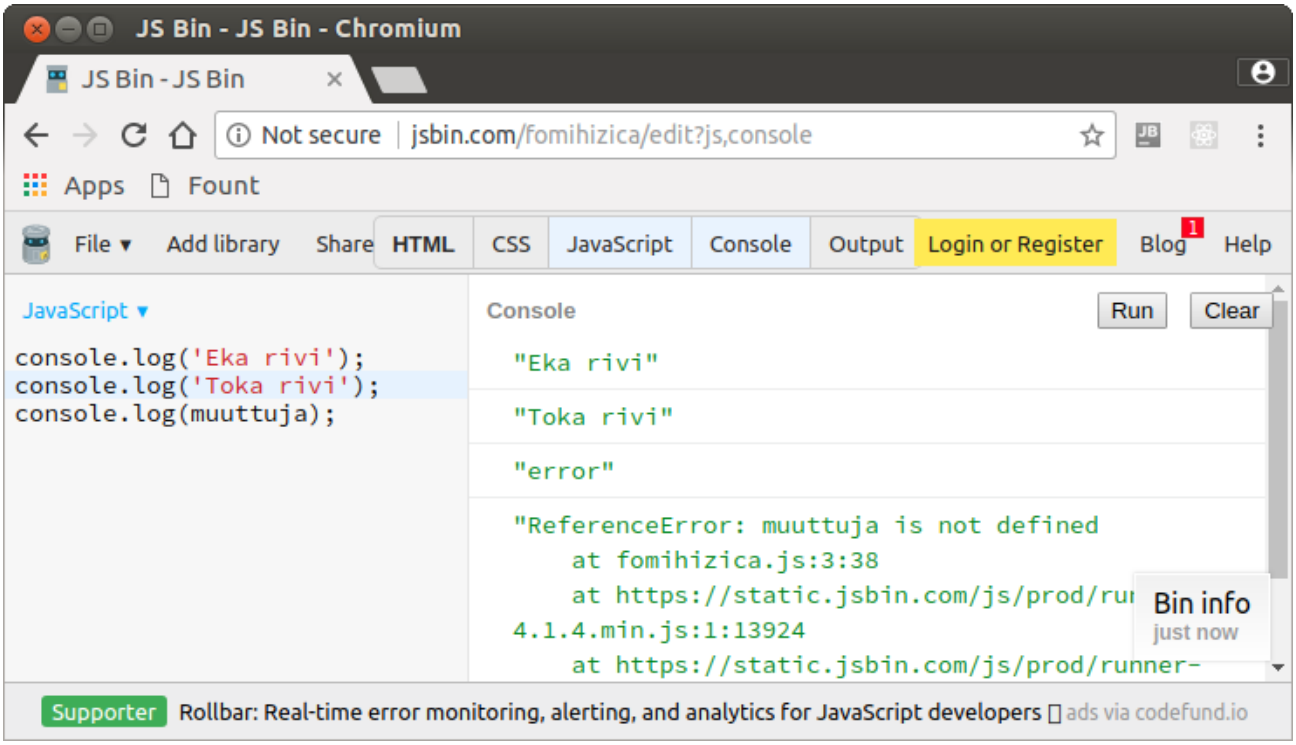
Esimerkki 0203

- Oheisessa kuvassa HTML-, JavaScript-, Console- ja Output-näkymät on aktivoitu vastaavilla tabeilla
- JS-koodia voidaan kirjoittaa sellaisenaan JavaScript-ikkunaan ja muutettu koodi voidaan ilman tallennuksia ja uudelleenlatauksia ajaa suoraan "Run with Js"-painikkeella



Esimerkki 0204

- Yksittäisten JS-asioiden kokeilemisessa riittää usein aktivoida vain JavaScript- ja Console-näkymät ja jättää HTML-osion BODY-elementtikin tyhjäksi. Näin kokeilu ja testaaminen yhdistettynä console.log()-metodin käyttämisen kanssa on todella suoraviivaista



Chrome-selaimen debuggerin käyttö

Chrome-selaimen debuggerin käytön mahdollisuuksiin kannattaa tutustua JavaScript-ohjelmoinnin opiskelussa mieluummin ennemmin kuin myöhemmin. Ohessa linkki pätevään aihealueen ohjeeseen:

- https://developers.google.com/web/tools/chrome-devtools/javascript

Muuttujat

- Muuttujat määritellään tavallisimmin var-avainsanaa käyttämällä
- muuttujiin voi tallentaa myös olioita

```
var mutu; // määrittely var-avainsanalla
var mutu = 500; // arvon sijoitus
var summa = a + b + c; // laskutoimitus
var t = [];  // uusi taulukko
```

Huom: ES6 mahdollistaa myös muuttujien määrittelyn let-avainsanalla. Siihen palataan

Globaalit muuttujat

Neljä tapaa määritellä

```
// var globaalissa näkyvyysalueessa
var title = "mjono";

// Piste-notaatio
```

```
window.title = "mjono";

// Olion ominaisuus hakasulku-notaatiolla
window.["title"] = "mjono";

// Funktion sisällä ilman var-avainsanaa:

function doSomething() {
  title = "mjono";
}
doSomething();
```

Taulukko

- JavaScriptissä taulukon alkioihin voi viitata **vain** *indeksinumeroilla*
- Hajautustaulukkoja (Associative arrays) **ei** ole
- Käytä olioita, kun haluat viitata alkioihin merkkijonoilla
- Taulukot voivat olla *useampiulotteisia* (array of arrays)
- Taulukkoa voi käsitellä erilaisilla *funktioilla* mm. pop, push, shift, sort.

Taulukon määrittely

Suoraan sijoittamalla

```
var kunnat = ["Laukaa", "Muurame", "Uurainen"];
```

Hakasulkumerkinnällä

```
t = []
t[0] = 10; // tai t.push(10);
t[1] = 11; // tai t.push(11);
t[2] = 12; // tai t.push(12);

// Alkion sijoittaminen taulukon loppuun
t[t.length] = 13; // tai t.push(13);

// Tulostetaan alkoiden lukumäärä
console.log(t.length); // tulostuu 4

// Lasketaan ja tulostetaan keskiarvo
let keskiarvo = (t[0]+t[1]+t[2]+t[3]) / t.length;
console.log(keskiarvo); // 11.5

// Taulukon arvojen tulostaminen indekseineen
for (let value of t) {
  console.log(t.indexOf(value) + ':' + value);
}
```

Toistorakenteet JavaScriptissä

JavaScript-kielessä voi käyttää seuraavia perinteisiä toistorakenteita

- while-lause

- do-while -lause
- for-lause
- for..in -lause
- for..of -lause. Uutena ES6-syntaksissa. Vastaa lähinnä monien kielien foreach-lausetta

Esimerkki 0205

`for..of`-lause:

```
let tulokset  = [1,2,3];

for(let arvo of tulokset) {
  console.log(arvo);
}

/* Tulostaa:
1
2
3
*/
```

JavaSript-perusteet, satunnaisia loppuhuomautuksia

- Merkkijonojen yhdistäminen eli katenointi tapahtuu +-operaattorilla

```
document.write(a + " ja " + b);
```

- Operaattori `===` vaatii yhtäsuuruuden lisäksi myös samaa tyyppiä

```
a = 1;
b = 1;
c = '1';

if (a == b)   // Tosi
if (a === b)  // Tosi
if (a == c)   // Tosi
if (a === c)  // Epätosi
```

Tunnuksissa kuten muuttujissa ja funktioiden nimissä kirjaintaso on merkitsevä (case-sensitive) eli `a` ja `A` ovat eri tunnuksia kuten vaikkapa funktiot `getvalue` ja `getValue`.